Intel® C++ Compiler 6.0 for Windows*
Getting Started Guide

Table of contents

1 Overview

This document explains how to install the Intel® C++ Compiler for Windows*, build "Hello World" in the Visual C++* 6.0 and Visual C++ .NET* environment for IA-32 and Itanium™-based systems; and how to get started optimizing your applications with the Intel® compilers.

The Intel C++ Compiler 6.0 for Windows consists of the following:

- Intel C++ Compiler for IA-32 based applications: `icl`
- Intel C++ Compiler for developing Itanium-based applications on IA-32 systems: `ecl`
- Intel C++ Compiler for Itanium-based applications: `ecl`
- IA-32-based Assembler to produce Itanium-based applications
- Itanium-based Assembler to produce Itanium-based applications
- Enhanced Debugger: `edb`
- Utilities
  - Package identification tool at the start menu
  - Programs\Intel(R) Software Development Tools\Registration and Support\Get Package ID.
  - Compiler selection tool that integrates Intel C++ Compiler 6.0 with Microsoft* Visual C++ 6.0 IDE
  - Compiler integration utilities that integrate Intel C++ Compiler 6.0 with Microsoft Visual C++ .NET IDE

- Utility to support non-administrative users at the start menu
  `Programs\Intel(R) Software Development Tools\Intel(R) C++ Compiler 6.0\Update User's Registry`.
- The Makefile Utility that provided users with the ability to switch between the Intel C++ Compiler 6.0 and the Microsoft Visual C++ 6.0 Compiler without requiring changes to their makefiles. The Makefile utility is available from Customer Installation Type only.
- The icpi Utility to isolate compile/link time errors. It is located at
  `<installation_directory>\Compiler60\ia32\bin\icpi.exe`
  or at
  `<installation_directory>\Compiler60\ia64\bin\icpi.exe`

- Product documentation

2 System Requirements

2.1 For developing applications on IA-32 systems

Minimum hardware requirements:

- A system based on an Intel® Pentium® processor, subsequent Pentium or Intel® Xeon™ processor for IA-32 applications; or a system with a 350 MHz Pentium II, Intel Xeon or greater processor for Itanium™-based applications
- 128 MB of RAM for IA-32 applications or 256 MB of RAM for Itanium-based applications
- 100 MB of disk space. During installation only, allow an additional 100 MB for the download and temporary intermediate files
- 100 MB of hard disk space for the virtual memory paging file. Be sure to use at least the minimum amount of virtual memory recommended by Microsoft* Windows*

Software requirements:

- Microsoft Windows 98*, Windows 98 SE*, Windows Millennium Edition*, Windows NT* 4.0 with SP 6 or higher, Windows 2000*, or Windows XP*.
  If you're developing with Microsoft Visual C++* .NET*, Windows NT 4.0 with SP6, or Windows 2000, or Windows XP is required
- Microsoft Visual C++ 6.0 Professional Edition or higher, or Microsoft Visual C++ .NET Professional Edition or higher. (Both Standard Editions are not supported). Microsoft Visual C++ .NET is part of Microsoft Visual Studio* .NET package
- Microsoft Platform SDK* is required to develop Itanium-based applications
- Microsoft Macro Assembler* (MASM) Version 6.14 or later is required if you want to use options that produce or operate on assembly files
- The Intel® C++ Compiler is not designed to be used in the Watcom* & Borland* development environments

The Intel C++ IA-32-based compiler for developing Itanium-based applications has been tested on the following environment when released:

- Microsoft Windows 2000 professional with service pack 2 with Platform SDK build 2462.1
- Microsoft Windows XP professional build 2600 with Platform SDK build 2601

2.2 For developing Itanium™-based applications on Itanium-based systems

Hardware requirements:

- A system with an Itanium processor or higher
- 512 MB of RAM (1 GB recommended)

- 100 MB free hard disk space. During installation only, allow an additional 100 MB for the download and temporary intermediate files

Software requirements:

- Microsoft Windows 2000* 64-bit edition, Windows XP* 64-bit edition or Windows Advanced Server*
- Microsoft Platform SDK*. The compiler has been tested using Microsoft Platform SDK
- NOTE: The DLL's in the Platform SDK directory `c:\Program files\Microsoft SDK\redist\win64` may also be required at runtime

The Intel C++ Itanium-based compiler has been tested on the following environment when released:

- Microsoft Windows 2000 64-bit edition (build 2462) with Platform SDK (build 2505)
- Microsoft Windows XP 64-bit edition (build 2600) with Platform SDK (build 2601)
- Microsoft Windows .NET Enterprise Server* 64-Bit Version 5.1 (Build 3590.main, 011110-1652) with Platform SDK (build 3590.1)

3 Installation Notes

The Intel® C++ Compiler 6.0 uses the Windows Installer*. This provides additional options for customization, update or repair of the installation, as well as providing a single option for uninstalling all components.

The Intel C++ Compiler uses the GlobeTrotter's* FLEXlm* electronic licensing technology. License management should be transparent. Please follow the installation steps below to install the FLEXlm license file.

NOTES: Before installing the compilers, please read the software requirements. Microsoft* Platform SDK* should be installed if developing Itanium™-based applications.

3.1 Install the Intel C++ compiler

1. Check the hardware and software requirements (see above for detail).
2. Obtain administrative (not power user) privilege that is needed in order to install the Intel C++ Compiler correctly.
3. Download the compiler package or insert the product CD-ROM.
4. Run the downloaded executable or setup.exe from CD-ROM and follow the setup program to finish the installation.
5. Install the FLEXlm license: For electronic download, the license is sent via email with install instructions. For the CD-ROM users, the installation program will install the corresponding license for you automatically to `c:\program files\Common Files\Intel\Licenses` On IA-32 systems or `c:\program files (x86)\Common Files\Intel\Licenses` on Itanium-based systems.
6. Use the Intel C++ compiler at a command prompt or through Microsoft Visual C++* 6.0 or Visual C++ .NET*.

3.2 Uninstalling/Repairing the Intel C++ Compiler

You need to obtain administrative (not power user) privilege in order to uninstall the Intel C++ Compiler correctly. There are two methods to uninstall or repair the Intel C++ Compiler.

- Use [Start->Programs->Intel(R) Software Development Tool->Modify or Remove Intel(R) C++ and EDB]
- Use Windows* [Control Panel->add/remove programs]

Caution: If you have developed programs with the Intel C++ Compiler through Microsoft Visual C++ .NET, please use slnconverter.exe to disassociate your solutions from the Intel C++ Compiler before uninstalling. See below for details about this utility.

4 Using the Intel C++ Compiler

A valid FLEXlm license should be installed before going forward.

4.1 Building "Hello World" with the Intel C++ Compiler

Building "Hello World" in command line

The following describes the steps to building the classic "Hello World" program.

1. Create a simple "Hello World" C++ program in a text editor "hello.cpp":

```
#include <iostream>
using namespace std;
int main()
{
     cout << "Hello World!" << endl;
     return 0;
}
```

2. Open a command window from [Start->Programs->Intel(R) Software Development Tools->Intel(R) C++ Compiler 6.0->Intel(R) C++ for 32-bit applications] to develop IA-32 applications. Or Open a command window from [Start->Programs->Intel(R) Software Development Tools->Intel(R) C++ Compiler 6.0->Intel(R) C++ for Itanium(TM) - based applications] to develop Itanium-based applications.
3. Compile hello.cpp:
   - **icl hello.cpp** ------ Creates an IA-32 application.
   - **ecl hello.cpp** ------ Creates an Itanium-based application
4. Run the executable: "**hello.exe**", it should display "Hello World!"

Building "Hello World" in Microsoft* Visual C++* 6.0

1. Open Microsoft Visual C++ 6.0
2. Create a Win32 Console Project named "hello", select "Hello World" application when creating the project
3. Open "Selection Tool" dialog from [Tools->Select Compiler] menu
4. Check "Intel C++ Compiler" inside group "IA-32 Compiler Selection" and click "Ok"
5. Build the project: you'll notice the Intel C++ Compiler "icl" is used in the output window
6. Run the executable to test

Building "Hello World" for Itanium-based systems in Microsoft Visual C++ 6.0

1. Follow step 1, 2, and 3 above
2. Check "Intel C++ Compiler" and "Use Environment Variables Listed Below" inside group "Intel Itanium Compiler/ Environment Selection", then click "Ok"
3. Open the "project settings" dialog
   a. Click on "C/C++" tab, add "WIN64" to "Processor definitions" input box.
   b. Click on "Link" tab, add "/machine:ia64" to "Project Options" input box.

c. Click "Ok"

4. Build the project: you'll notice the Intel C++ Compiler "ecl" for Itanium-based systems is used in the output window
5. Run the executable to test

Building "Hello World" for IA-32 systems with the Intel C++ Compiler in Microsoft Visual C++ .NET*

1. Create a C++ Win32 project "hello", in "Application Settings" tab select "Console application" and click "Finish" button
2. Open "hello.cpp", add the following:
   a. Add to top:
   ```
   #include <iostream>
   using namespace std;
   ```
   b. Add to "main()":
   ```
   cout << "Hello World!" << endl;
   ```
3. Open [Tools->Options] dialog, click on "Intel Integration Tool" item. Select a version of Intel C++ Compiler you'd like to use if more than one Intel C++ Compiler 6.0 or higher installed and click "Ok".
4. Open "hello Property Pages" dialog, click on "Intel Specific", from the "Compiler name" drop-down box, select "Intel C++ Compiler (icl)" and click "Ok"
5. Build the solution
6. Run the executable to test

Building existing Visual C++ 6.0 "Hello World" for IA-32 systems with the Intel C++ Compiler in Microsoft Visual C++ .NET

1. Convert the exiting "Hello World" project into a solution with Microsoft Visual C++ .NET
2. Associate the "Hello World" solution with Intel Compiler Integration using following command from the supplied Intel C++ Compiler Command Prompts at [Start->Programs->Intel(R) Software Development Tools->Intel(R) C++ Compiler 6.0->Intel(R) C++ for 32-bit applications]:
   ```
   >>SlnConverter MyHello.sln Intel
   ```
3. Open the solution with Visual C++ .NET
4. Follow the step 3, 4 and 5 above to build the solution

Using Intel C++ Compiler in Batch-mode with Microsoft Visual C++ .NET

To support batch-mode execution of Visual C++ .NET, the Intel Integration Tool adds the extra command line option **/IntelSpecific**. This option may be used to select the Intel C++ compiler or the Microsoft Visual C++ compiler during command batch-mode execution. You should run this command from the supplied Intel C++ Compiler Command Prompts from [Start->Programs->Intel(R) Software Development Tools->Intel(R) C++ Compiler 6.0->Intel(R) C++ for 32-bit applications].

The syntax of the batch-mode invocation for Visual C++ .NET is:

```
devenv.exe solutionfile.sln /IntelSpecific compilername [compilerversion]
/build solutionconfig [/project projectnameorfile [/projectconfig name]]
```

Where:

- **/IntelSpecific** is an option of the Intel C++ Integration Tool,

- **compilername** is a parameter to the /IntelSpecific option. It can have the value **Intel** or **Microsoft.**
- **compilerversion** is an optional parameter to the /IntelSpecific option. It specifies the name of an installed Intel compiler (for example "Intel C++ 6.0").
  This option is not useful unless multiple Intel compiler versions are installed.

Example:

devenv.exe d:\test\qwer7\qwer7.sln /build debug /IntelSpecific Intel

NOTE:
Building applications for Itanium-based systems with the Intel C++ Compiler in Microsoft* Visual C++ .NET* is not supported.

4.2 Building only one file in a project with the Intel C++ Compiler

1. Open your project with Visual C++ 6.0 or Visual C++ .NET
2. Right click on the file you want to build with the Intel C++ Compiler, select "settings" in Visual C++ 6.0 or "properties" in Visual C++ .NET
3. Add "`_USE_INTEL_COMPILER`" to "`preprocessor definitions`", then click "`Ok`"
4. Make sure "`Intel C++ Compiler`" is not selected as the compiler for the project/solution
   a. In Visual C++ 6.0, use the "`Selection Tool`" to verify
   b. In Visual C++ .NET, use the "`Property pages`" to verify
5. Build the project: you'll notice the Intel C++ Compiler is used for the file in the output window
6. Run the executable to test

4.3 Utilities in Intel C++ Compiler

Utility to enable/ disable Intel C++ Compiler Integration from Microsoft Visual C++ .NET
The program `RegIntelPkg.exe` provides the easiest way to disable the Intel C++ Compiler without uninstalling. You can invoke this program from `[Start->Programs->Intel(R) Software Development Tools->Intel(R) C++ Compiler 6.0->Intel(R) C++ Integration Tool->Enable/Disable Intel C++ Integration]`

Utility to associate/ disassociate Intel C++ Compiler integration to/from your solutions
The utility `SlnConverter.exe` is used to associate/ disassociate Intel C++ Compiler integration with a Visual C++ .NET solution.

- For solutions that are created before installing the Intel C++ Compiler, you need to run `slnconverter.exe` to associate them with Intel C++ Compiler integration. Then you can use the above methods to compile with the Intel C++ Compiler in Visual C++ .NET IDE.
- Before uninstalling the Intel C++ Compiler, please use this utility to disassociate the Intel C++ Compiler integration with solutions that have been compiled by the Intel C++ Compiler within Visual C++ .NET IDE.
- If you have a VC++ 6.0 project and you'd like to compile it with the Intel C++ Compiler within Visual C++ .NET, please use Visual C++ .NET to convert it into a solution first; then use this utility to associate it with Intel C++ Compiler integration.
- Syntax:
  - **`SlnConverter myHelloWorld.sln Intel`**: to associate the solution with Intel C++ Compiler integration

- **`SlnConverter myHelloWorld.sln Microsoft`**: to disassociate the solution with Intel C++ Compiler integration.

Utility to enable regular user other than administrative user to use the Intel selection tool within Visual C++* 6.0
This utility is located at `[Start->Programs->Intel(R) Software Development Tools->Intel(R) C++ Compiler 6.0->Update User's Registry]`. For a regular user to use the Intel selection tool within Visual C++ 6.0 IDE, you need to run this utility first.

4.4 Compatibility with Visual C++ .NET

The Intel C++ Compiler 6.0 supports limited features of Visual C++ .NET, and does not support managed code.
For detailed information, please refer to "**Intel C++ compiler 6.0 and Microsoft Visual C++ .NET compatibility**" at http://www.intel.com/software/products/compilers/c60/.
For more information on using Intel C++ Compiler 6.0 within Visual C++ .NET, please see "**Using the Intel® C++ Compiler with Microsoft* Visual C++* .NET***" section in the Intel C++ Compiler's Compiler User's Guide at `<install-dir>/compiler60/docs/ccug.chm`.

5 Getting Started with Intel Compiler Optimizations

The Intel C++ Compiler enables programmers to take full advantage of the advanced performance enhancement features of Intel's latest IA-32 and Itanium processors and includes advanced optimizations. These include support for Streaming SIMD Extensions 2, profile-guided optimization, interprocedural optimization, vectorization and processor dispatch.

The optimizations are intended for use in product-release builds of applications, not necessarily for earlier phases of application development cycles. In general, increasing the degree of optimization done by the compiler leads to an increase in compile-time and reduced debugging capability. This section describes an optimization methodology with the Intel C++ Compiler.

During the application development, the "`-Zi -Od`" switches are recommended to allow fast compile times and full debugging with no optimization. To start to optimize, the default optimization "`-O2`" is recommended. The "`-O3`" option enables advanced optimizations. **Interprocedural optimization** allows the compiler to optimize across different compilation units and can have large performance improvements. **Profile guided optimization** uses information obtained by running an instrumented executable that allows the compiler to rebuild the application knowing where the majority of the computations are. Of course, not all optimizations are beneficial for all applications. For additional details on optimizing, the paper, "**Optimizing Applications with the Intel C++ and Fortran Compilers**" is available at http://www.intel.com/software/products/compilers/C60/. For complete information on the individual optimizations, please refer to Intel C++ Compiler's Compiler User's Guide at <install-dir>/compiler60/docs/ccug.chm.

Remember to always measure the performance of your application after each optimization added to verify the benefits. The VTune™ Performance Analyzer can be a great help for measuring the performance benefits of each, as well as giving advice on further tuning opportunities. Additional information is available at http://www.intel.com/software/products/vtune/.

6 Additional Information

Intel® Premier Support web site: Your feedback is very important to us. To receive technical support and product updates for the tools provided in this product you need to register at `http://support.intel.com/support/performancetools/c/v6/windows/`. To submit an issue or feature request, please go to Intel Premier Support at https://premier.intel.com/ after registering.

Compiler support information: Top technical issues and product errata are available at `http://support.intel.com/support/performancetools/c/v6/windows/`

Product release notes: Located at `<install-dir>/compiler60/docs/Crelnotes.htm`

Compiler User's Guide: Located at `<install-dir>/compiler60/docs/ccug.chm`

7 Copyright and Legal Information